URL: http://www.elsevier.nl/locate/entcs/volume30.html 1 pages

Linearity Analysis of Concurrent Logic Programs

Kazunori Ueda [1]

Department of Information and Computer Science
Waseda University
3–4–1, Okubo, Shinjuku-ku, Tokyo 169-8555, Japan

Abstract

Automatic memory management and the hiding of the notion of pointers are the prominent features of symbolic processing languages. They make programming easy and guarantee the safety of memory references. For the memory management of linked data structures, copying garbage collection is most widely used because of its simplicity and desirable properties. However, if certain properties about runtime storage allocation and the behavior of pointers can be obtaind by static analysis, a compiler may be able to generate object code closer to that of procedural programs. In the fields of parallel, distributed and real-time computation, it is highly desirable to be able to identify data structures in a program that can be managed without using garbage collection. To this end, this paper proposes a framework of linearity analysis for a concurrent logic language Moded Flat GHC, and proves its basic property. The purpose of linearity analysis is to distinguish between fragments of data structures that may be referenced by two or more pointers and those that cannot be referenced by two or more pointers. Data structures with only one reader are amenable to compile-time garbage collection or local reuse. The proposed framework of linearity analysis is constraint-based and involves both equality and implicational constraints. It has been implemented as part of klint v2, a static analyzer for KL1 programs.

[1] A full version of this work will appear in *Proc. International Workshop on Parallel and Distributed Computing for Symbolic and Irregular Applications*, World Scientific Publishing. The work is partially supported by Grant-In-Aid ((A)(1)09245101 and (C)(2)11680370) for Scientific Research, Ministry of Education, and Waseda University Grant (98A-575) for Special Research Projects.